Two-Phased Real-Time Rendering of Large Neuron Databases

Pablo de Heras Ciechomski, Robin Mange and Achille Peternier
Visualbiotech
PSE-C EPFL, 1015 Ecublens, Switzerland
www.visualbiotech.ch

pablo@visualbiotech.ch, robin@visualbiotech.ch, achille@viusalbiotech.ch

Keywords: computer graphics, visualization, computational biology, neuroinformatics

Abstract: By dividing the rendering of neurons into two modes called exploration mode and static mode, the user can explore large neuron databases interactively. The *exploration* mode contains only the necessary visual characteristics of neurons for the user to be able to navigate inside of the circuit, with an upper limit of primitives used. It is done in a forward rendering fashion. The *static* mode is of a higher visual quality, where neurons are rendered with a deferred shading technique, achieving a constant time update frequency.

1 Introduction

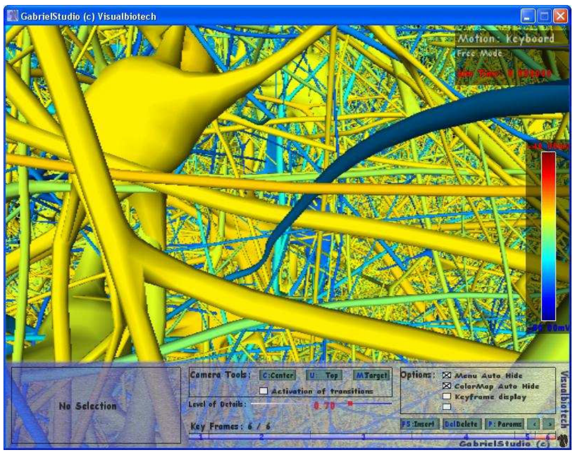

Figure 1: Selecting a neuron in GabrielStudio. Raw neuron data courtesy the Blue Brain Project. Copyright Visualbiotech 2006-2008, all rights reserved.

This paper presents a method for rendering large amounts of three dimensional neurons in real-time in the software GabrielStudio by Visualbiotech (Visualbiotech, 2008). The user can travel inside of the data which describes neuronal circuits, with full six degrees of freedom (6 DOF), similar to a virtual microscope. When the user is moving continuously the mode is called *exploration* and when he stands still it is called *static*. These two modes switch from one to the other on the starting or stopping of camera movement. Thus a user can explore large neuron circuits interactively and enjoy a highly detailed view of them when standing still.

This approach differs from (de Heras Ciechomski and Mange, 2008) in that no longer is a static octree employed, where all data such as triangles and individual neuron segments must be duplicated. Instead neurons are treated like a ”soup” of instances derived from templates, that in turn can move or rotate in any direction. This aids in reducing memory costs, which quickly can become prohibitive with traditional grid approaches. There is no prior building time to loading a circuit, thus no waiting is needed. Aside from these clear benefits this paper presents two different ways of rendering that compliment each other.

In addition to the two rendering modes, neurons are treated with a continuous per primitive, projected level of detail and an approximate (sub-pixel accurate) anti-aliasing method. The level of detail model refers to a hierarchic rendering representation, opposed to the more common triangle reduction LOD approach, see (Luebke et al., 2002) and (Schaufler and Stürzlinger, 1995). Levels of detail are not merely data reductions, instead they reflect the visual appearance of neurons.

2 Exploration Mode

An exploration mode is engaged whenever the user is turning the point of view of the virtual microscope. Its purpose is to be scalable in terms of rendering speed, benefiting a real-time exploration of a large neuronal circuit. The rendering method utilized is called forward rendering, which means that each pixel on the screen has been fully evaluated at the time it is rendered, so that no extra processing is needed when the image is composited. Most graphics cards work best in a batched rendering mode, where reading back results from the graphics card to the system memory is too slow to be used for real-time purposes, as it stalls the graphics pipeline, see (Haines, 2006).

2.1 Limiting Primitives

The smallest graphics primitive on a graphics accelerator card is usually a triangle or a point (which on most hardware is emulated using a triangle), and a graphics card can handle a set amount of triangles or pixels painted. Most geometric scenes are triangle or vertex bound, which is why the primitive counter in the presented solution is set to be individual triangles or quadrilaterals. A scene consists of neurons, which in turn individually consist of the soma or center point of the body, which branches out into thick dendrites that in turn branch out into thinner and thinner dendrites. When representing a neuron the soma is usually the most important to show, then the thicker or more visible dendrites and lastly thinner dendritic branches.

The central feature of the problem is to distribute the set limit amount of graphical primitives, on the most important neurons and dendrites, see figure 2. A brute force method is to set the representational score for all dendritic parts of neurons and sort them accordingly. Then render the sorted list until the set limit amount is reached. This is all well, as long as the number of neurons and dendrites is low, but since dendritic parts quickly become numerous it is a limiting factor, see figure 3. The chosen solution uses, albeit not optimal in terms of quality, instead to sort on the soma center point and to test dendrites of nearby neurons first.

2.2 Dendritic Sampling LOD

Instead of using a distance measure to decide if a neuron or part of a neuron is visible, the size in pixels of the hierarchy is the deciding factor. The closer to the viewer, the bigger the size in pixels of the part and the bigger the chance it will be visible. If parts of neurons are drawn as triangles, aliasing will occur when said parts (often dendrites) become less than a pixel wide. To ensure this does not occur, GabrielStudio renders all sub-pixel parts as one pixel wide lines, thus removing the sub-pixel precision problems. This does lead to aliasing in the opposite direction, such that parts that should be invisible become visible, however from a point of view of neuroinformatics it is better over-represent this information.

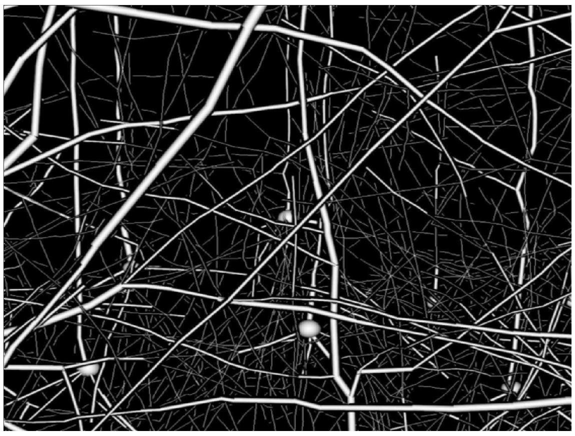
Figure 2: Neurons are represented as diameter thick lines or segments and lines. An ambient occlusion pass is added to emphasize the depth. (c) 2007 copyright Visualbiotech, all rights reserved.

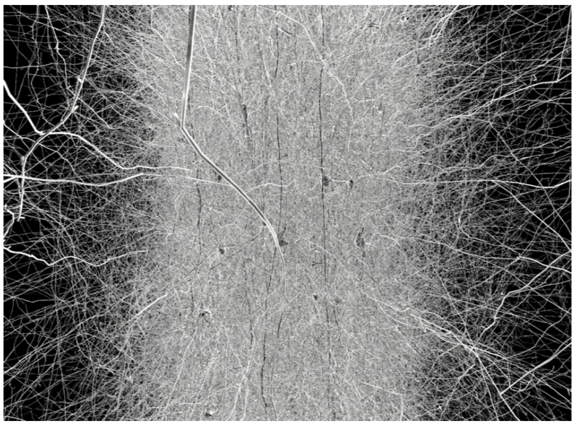
Figure 3: The complexity of the neuron forest quickly grows with the number of dendrites shown on screen. Copyright 2007 Visualbiotech, all rights reserved.

A dendrite consists of a string of points with diameters such that they describe volumetric tubes. Each pair of points on the dendrite is treated separately. When a tube between the points becomes too thin, it is represented using a pixel wide line and when thicker as an oriented quadrilateral, see (de Heras Ciechomski and Mange, 2008) and (Stoll et al., 2005). The number of points used to represent the dendrite itself, are dependent on the projected size of the dendrite such that when bigger, the more samples are used. This ensures that curved dendrites are treated properly when closer to the viewer.

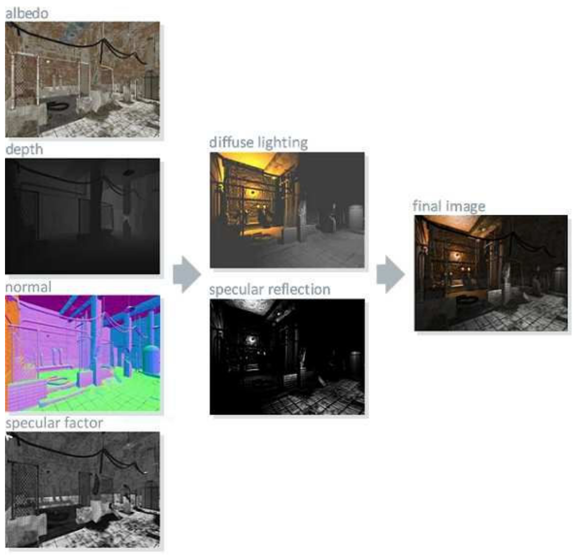

Figure 4: An example overview of different passes applied to a deferred shading pipeline. (Leadwerks, 2008).

3 Static Mode

When the user is engaged in the static mode, the camera does not move but the user can select neurons to be clamped and watch simulation data replays on high detail mesh models of neurons. The first phase of the static mode is the building of the deferred rendering G-buffer (see (Shishkovtsov, 2005)) which holds all the data necessary for the pixel to be evaluated such as color, lighting, textures and simulation values. It is the most costly phase and requires that the user waits a few seconds before the high resolution image appears. After the initial seeding phase the deferred rendering pipeline takes over.

3.1 Deferred Shading

Deferred shading or rendering, entails evaluating lighting computations, such as surface shading and simulation coloring at a later stage. The opposite of deferred shading is called forward rendering and involves compositing each pixel of the final image disregarding visibility of said fragment. In a deferred pipeline only visible pixels are treated. The final color is calculated for the red component in this example as

$$diff = max(dot(normal, light), 0) \tag{1}$$
$$color_r = texture_r * sim_r * diff \tag{2}$$
$$color_r = color_r(0.5 + exp(diff, lumin)) \tag{3}$$

This formula can be simplified to

$$color_r = sim_r * surface_r \tag{4}$$

Using the above short form one can separate the simulation color update of the neuron to a per visible pixel multiplication, given that the other parameters are known. In deferred shading the above formulas are sometimes called the G-buffer. Since textures are grey scale the simulation color is used to indicate a selected neuron or a simulation color.

3.2 Constant Time Rendering

Employing the deferred shading pipeline the user knows exactly the visible pixels, which neuron, dendrite and part of dendrite the pixels belong to and their separable surface constant. The simulation data base can now be queried using the exact amount of simulation values, which is important as simulation values per frame, range in the hundreds of millions for a circuit of 10’000 neurons. On average a few thousand to a few hundred thousand simulation values are required for the visible pixels. This a reduction of a minimum 1000 times up to 100’000 times of the simulation data required to read from the database.

Not only is the simulation query and its application constant in time, but also its rendering. The update frequency is normally in the hundreds of frames per second on an average desktop PC.

3.3 Special Effects

Since the rendering is almost for free in the static mode several special effects are added on top of the graphics buffer such as screen space ambient occlusion as seen in figures 6 and 2. It works by measuring nearby depth values averaging them and writing them as a shading value for that pixel, see (Tarini et al., 2006).

4 Results

The following results are acquired on a standard laptop with 2 GB internal memory, an Nvidia graphics

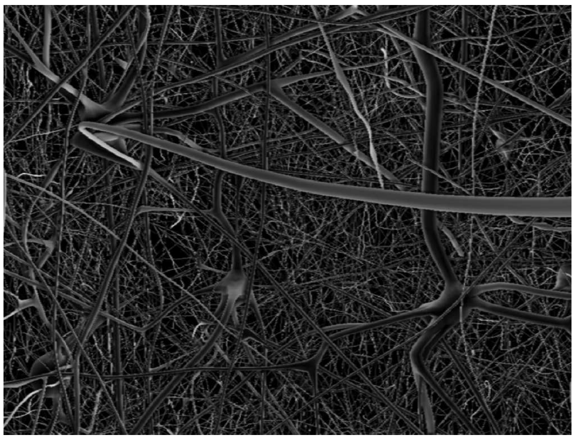
Figure 5: An example of deferred rendering in GabrielStudio with a screen space ambient occlusion effect. Close up neurons are represented as meshes. Copyright 2007 Visualbiotech, all rights reserved.

accelerator card with 256 MB memory and a dual core Intel processor running Windows XP. It is a standard computer from 2006, to compare with the massive parallel workstation of 32 processors, 300 GB and 16 graphics cards of (de Heras Ciechomski and Mange, 2008).

The free exploration mode is run with a limit of 100'000 primitives in total, where the scene consists of 10'000 neurons, and where only a fraction of all dendrites are visible. Since the LOD system is based on projected size, the most visible ones are drawn first. The dendrites are not smooth when seen from far and are smooth when close up, because of the point sampling rate increase. The user can choose with a slider the rate of projected pixel importance. Another slider sets the rate of dendritic point sampling. The minimum frames per second is 20 and on average it is 30.

When the user stops and clicks on the static mode, the last frame of the interactive mode is continuously displayed and its simulation values are updated. However, it can last several seconds, depending on scene complexity and if all neurons are chosen to be displayed as meshes. Most of the time users prefer the soma, segment and line model with more primitives (1 million instead of 100'000) instead of a mesh based representation for the static mode. The speed of the static mode is around 100 to 120 frames per second.

The screen space ambient occlusion helps with depth queues for large complex scenes, but is not good for simulation data as the shading values will alter the simulation scale values.

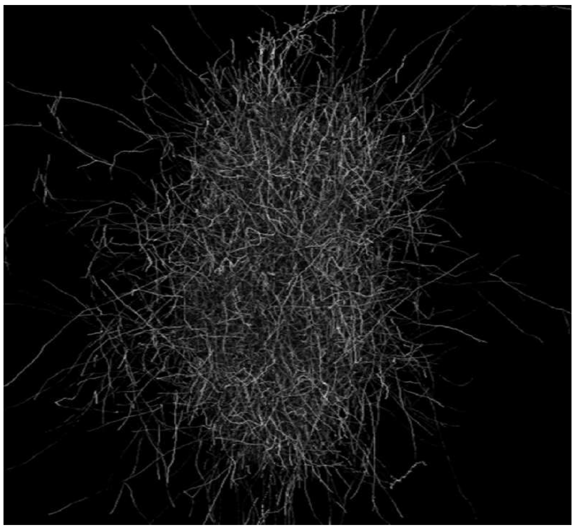
Figure 6: An example of deferred rendering in GabrielStudio with a screen space ambient occlusion effect. Copyright 2007 Visualbiotech, all rights reserved.

5 Conclusion

Splitting rendering into an exploration mode and a static phase, helps in terms of requiring less processing for the interactive real-time mode and a very quick and highly detailed static mode. The current initialization of the static mode (while the last exploration frame is displayed) can take several seconds, which could be improved with an adaptive algorithm where the amount of primitives is constrained between updates, giving the impression to the user of a steadily increasing quality as time passes.

Limiting the primitives of the exploration mode is essential as without it is not possible to navigate through the neuron forest in real-time.

Acknowledgements

Blue Brain Project for raw neuron data.

REFERENCES

de Heras Ciechomski, P. and Mange, R. (2008). Realtime neocortical column visualization. In *BIOSIGNALS (2)*, pages 283–288.

Haines, E. (2006). An introductory tour of interactive rendering. *IEEE Computer Graphics and Applications*, 26(1):76–87.

Leadwerks (2008). Leadwerks software, usa, www.leadwerks.com.

Luebke, D., Watson, B., Cohen, J. D., Reddy, M., and Varshney, A. (2002). *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA.

Schaufler, G. and Stürzlinger, W. (1995). Generating multiple levels of detail from polygonal geometry models. In Göbel, M., editor, *Virtual Environments '95 (Eurographics Workshop)*, pages 33–41. Springer-Verlag: Heidelberg, Germany.

Shishkovtsov, O. (2005). Chapter 9. deferred shading in s.t.a.l.k.e.r. *GPU Gems 2 : Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*.

Stoll, C., Gumhold, S., and Seidel, H.-P. (2005). Visualization with stylized line primitives. In *IEEE Visualization*, page 88.

Tarini, M., Cignoni, P., and Montani, C. (2006). Ambient occlusion and edge cueing for enhancing real time molecular visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1237–1244.

Visualbiotech (2008). Gabrielstudio (tm), a visualization library for biotechnology, www.visualbiotech.ch.